

## SCENE REPRESENTATION METHOD AND SYSTEM

## TECHNICAL FIELD OF THE INVENTION

This invention relates generally to image processing and more particularly to a scene representation method and system.

BACKGROUND OF THE INVENTION

Programmable algorithms such as shading procedures typically require access to parameters that may be associated with a geometry to be drawn or rendered for an image scene or an object therein. Generally, the values of these parameters vary over each surface in response to an artist or technical director's input. For example, an artist may desire that colors vary across a portion of an image scene or object within. Algorithms such as shaders available to artists are thus desirably programmable to increase the flexibility with which designs may be implemented and/or modified.

Many systems provide a variety of hardware-accelerated methods to implement sophisticated surface shading algorithms to increase processing speeds. These conventional systems may take advantage of a graphics systems interface such as OPENGGL® and/or a graphics pipeline. However, some of these systems typically break down programmable algorithms into steps that may be executed by the graphics systems interface and/or the graphics pipeline, but do not supply the parameters with geometry data or allow the parameters to be evaluated on the surface. Other systems may accept parameters but do not associate or evaluate them with tessellated surface values. As a result, these conventional systems typically cannot take advantage of the graphics pipeline, and thus may suffer from increased processing time. As a consequence, an artist's flexibility to change surface appearances interactively by using the parameters is reduced.

In addition, those systems that do utilize the graphics pipeline typically break down the algorithms into steps so specific to the pipeline that they cannot maintain the same level of performance while providing portability from platform to platform. In addition, some of these systems typically do not manage data to be used to draw image scenes with multiple objects using different programmable algorithms. Such a configuration may unnecessarily replicate data, exceed hardware resources and/or interfere with rendering of various objects.

SUMMARY OF THE INVENTION

From the foregoing, it may be appreciated that a need has arisen for a method and system for supplying and managing scene parameters for programmable shading applications. In accordance with the present invention, a system and method are provided that substantially reduce or eliminate disadvantages and problems of conventional systems.

One aspect of the invention is a method for representing a scene. The method includes providing a higher-level appearance description of an appearance of geometry in a retained-mode representation. The method also includes traversing the retained-mode representation to provide a final representation that can be rendered by a graphics pipeline.

The invention provides several important advantages. Various embodiments of the invention may have none, some, or all of these advantages. For example, the invention may be used to provide an infrastructure between a descriptive programmable application and a graphics system interface such as OpenGL®. For example, one technical advantage of the invention allows traversal of a higher-level appearance description so that it is translated into another representation that may be drawn for a graphics pipeline. Another technical advantage of the present invention is that it allows access to parameters for geometry that may be defined on any parametric surface. These parameters may then be evaluated on the surface by a graphics system interface for use in a graphics hardware or acceleration pipeline. The traversal may be implemented as a procedure using a variety of methods, and may be used with a variety of existing systems. For example, the invention may be integrated into an existing system library. In addition, the invention may also allow for extensibility so that new algorithms may be added as needed.

Another technical advantage of the present invention is that it may be used to manage data. For example, the invention may also minimize data replication and/or optimize the order of steps to be performed. Such an advantage may reduce the possibility that resources available may be exceeded, and may also minimize processing time and resources used. In addition, the invention may cull or identify various levels of detail needed to represent the appearance of a scene. Such an advantage may also reduce memory requirements and/or the possibility of

5

interference between rendering of a variety of objects. For example, each traversal may manage data variables that may be shared, which may reduce or minimize an amount of data sent through a graphics pipeline, thus freeing up resources and bandwidth. This allows graphics systems interfaces such as OPENGL® to be used with graphics pipelines in a multi-pass scenario. Other technical advantages may be readily ascertainable by those skilled in the art from the following figures, description, and claims.

FIG. 1  
FIG. 2  
FIG. 3  
FIG. 4  
FIG. 5  
FIG. 6  
FIG. 7  
FIG. 8  
FIG. 9  
FIG. 10  
FIG. 11  
FIG. 12  
FIG. 13  
FIG. 14  
FIG. 15  
FIG. 16  
FIG. 17  
FIG. 18  
FIG. 19  
FIG. 20  
FIG. 21  
FIG. 22  
FIG. 23  
FIG. 24  
FIG. 25  
FIG. 26  
FIG. 27  
FIG. 28  
FIG. 29  
FIG. 30  
FIG. 31  
FIG. 32  
FIG. 33  
FIG. 34  
FIG. 35  
FIG. 36  
FIG. 37  
FIG. 38  
FIG. 39  
FIG. 40  
FIG. 41  
FIG. 42  
FIG. 43  
FIG. 44  
FIG. 45  
FIG. 46  
FIG. 47  
FIG. 48  
FIG. 49  
FIG. 50  
FIG. 51  
FIG. 52  
FIG. 53  
FIG. 54  
FIG. 55  
FIG. 56  
FIG. 57  
FIG. 58  
FIG. 59  
FIG. 60  
FIG. 61  
FIG. 62  
FIG. 63  
FIG. 64  
FIG. 65  
FIG. 66  
FIG. 67  
FIG. 68  
FIG. 69  
FIG. 70  
FIG. 71  
FIG. 72  
FIG. 73  
FIG. 74  
FIG. 75  
FIG. 76  
FIG. 77  
FIG. 78  
FIG. 79  
FIG. 80  
FIG. 81  
FIG. 82  
FIG. 83  
FIG. 84  
FIG. 85  
FIG. 86  
FIG. 87  
FIG. 88  
FIG. 89  
FIG. 90  
FIG. 91  
FIG. 92  
FIG. 93  
FIG. 94  
FIG. 95  
FIG. 96  
FIG. 97  
FIG. 98  
FIG. 99  
FIG. 100  
FIG. 101  
FIG. 102  
FIG. 103  
FIG. 104  
FIG. 105  
FIG. 106  
FIG. 107  
FIG. 108  
FIG. 109  
FIG. 110  
FIG. 111  
FIG. 112  
FIG. 113  
FIG. 114  
FIG. 115  
FIG. 116  
FIG. 117  
FIG. 118  
FIG. 119  
FIG. 120  
FIG. 121  
FIG. 122  
FIG. 123  
FIG. 124  
FIG. 125  
FIG. 126  
FIG. 127  
FIG. 128  
FIG. 129  
FIG. 130  
FIG. 131  
FIG. 132  
FIG. 133  
FIG. 134  
FIG. 135  
FIG. 136  
FIG. 137  
FIG. 138  
FIG. 139  
FIG. 140  
FIG. 141  
FIG. 142  
FIG. 143  
FIG. 144  
FIG. 145  
FIG. 146  
FIG. 147  
FIG. 148  
FIG. 149  
FIG. 150  
FIG. 151  
FIG. 152  
FIG. 153  
FIG. 154  
FIG. 155  
FIG. 156  
FIG. 157  
FIG. 158  
FIG. 159  
FIG. 160  
FIG. 161  
FIG. 162  
FIG. 163  
FIG. 164  
FIG. 165  
FIG. 166  
FIG. 167  
FIG. 168  
FIG. 169  
FIG. 170  
FIG. 171  
FIG. 172  
FIG. 173  
FIG. 174  
FIG. 175  
FIG. 176  
FIG. 177  
FIG. 178  
FIG. 179  
FIG. 180  
FIG. 181  
FIG. 182  
FIG. 183  
FIG. 184  
FIG. 185  
FIG. 186  
FIG. 187  
FIG. 188  
FIG. 189  
FIG. 190  
FIG. 191  
FIG. 192  
FIG. 193  
FIG. 194  
FIG. 195  
FIG. 196  
FIG. 197  
FIG. 198  
FIG. 199  
FIG. 200  
FIG. 201  
FIG. 202  
FIG. 203  
FIG. 204  
FIG. 205  
FIG. 206  
FIG. 207  
FIG. 208  
FIG. 209  
FIG. 210  
FIG. 211  
FIG. 212  
FIG. 213  
FIG. 214  
FIG. 215  
FIG. 216  
FIG. 217  
FIG. 218  
FIG. 219  
FIG. 220  
FIG. 221  
FIG. 222  
FIG. 223  
FIG. 224  
FIG. 225  
FIG. 226  
FIG. 227  
FIG. 228  
FIG. 229  
FIG. 230  
FIG. 231  
FIG. 232  
FIG. 233  
FIG. 234  
FIG. 235  
FIG. 236  
FIG. 237  
FIG. 238  
FIG. 239  
FIG. 240  
FIG. 241  
FIG. 242  
FIG. 243  
FIG. 244  
FIG. 245  
FIG. 246  
FIG. 247  
FIG. 248  
FIG. 249  
FIG. 250  
FIG. 251  
FIG. 252  
FIG. 253  
FIG. 254  
FIG. 255  
FIG. 256  
FIG. 257  
FIG. 258  
FIG. 259  
FIG. 260  
FIG. 261  
FIG. 262  
FIG. 263  
FIG. 264  
FIG. 265  
FIG. 266  
FIG. 267  
FIG. 268  
FIG. 269  
FIG. 270  
FIG. 271  
FIG. 272  
FIG. 273  
FIG. 274  
FIG. 275  
FIG. 276  
FIG. 277  
FIG. 278  
FIG. 279  
FIG. 280  
FIG. 281  
FIG. 282  
FIG. 283  
FIG. 284  
FIG. 285  
FIG. 286  
FIG. 287  
FIG. 288  
FIG. 289  
FIG. 290  
FIG. 291  
FIG. 292  
FIG. 293  
FIG. 294  
FIG. 295  
FIG. 296  
FIG. 297  
FIG. 298  
FIG. 299  
FIG. 300  
FIG. 301  
FIG. 302  
FIG. 303  
FIG. 304  
FIG. 305  
FIG. 306  
FIG. 307  
FIG. 308  
FIG. 309  
FIG. 310  
FIG. 311  
FIG. 312  
FIG. 313  
FIG. 314  
FIG. 315  
FIG. 316  
FIG. 317  
FIG. 318  
FIG. 319  
FIG. 320  
FIG. 321  
FIG. 322  
FIG. 323  
FIG. 324  
FIG. 325  
FIG. 326  
FIG. 327  
FIG. 328  
FIG. 329  
FIG. 330  
FIG. 331  
FIG. 332  
FIG. 333  
FIG. 334  
FIG. 335  
FIG. 336  
FIG. 337  
FIG. 338  
FIG. 339  
FIG. 340  
FIG. 341  
FIG. 342  
FIG. 343  
FIG. 344  
FIG. 345  
FIG. 346  
FIG. 347  
FIG. 348  
FIG. 349  
FIG. 350  
FIG. 351  
FIG. 352  
FIG. 353  
FIG. 354  
FIG. 355  
FIG. 356  
FIG. 357  
FIG. 358  
FIG. 359  
FIG. 360  
FIG. 361  
FIG. 362  
FIG. 363  
FIG. 364  
FIG. 365  
FIG. 366  
FIG. 367  
FIG. 368  
FIG. 369  
FIG. 370  
FIG. 371  
FIG. 372  
FIG. 373  
FIG. 374  
FIG. 375  
FIG. 376  
FIG. 377  
FIG. 378  
FIG. 379  
FIG. 380  
FIG. 381  
FIG. 382  
FIG. 383  
FIG. 384  
FIG. 385  
FIG. 386  
FIG. 387  
FIG. 388  
FIG. 389  
FIG. 390  
FIG. 391  
FIG. 392  
FIG. 393  
FIG. 394  
FIG. 395  
FIG. 396  
FIG. 397  
FIG. 398  
FIG. 399  
FIG. 400  
FIG. 401  
FIG. 402  
FIG. 403  
FIG. 404  
FIG. 405  
FIG. 406  
FIG. 407  
FIG. 408  
FIG. 409  
FIG. 410  
FIG. 411  
FIG. 412  
FIG. 413  
FIG. 414  
FIG. 415  
FIG. 416  
FIG. 417  
FIG. 418  
FIG. 419  
FIG. 420  
FIG. 421  
FIG. 422  
FIG. 423  
FIG. 424  
FIG. 425  
FIG. 426  
FIG. 427  
FIG. 428  
FIG. 429  
FIG. 430  
FIG. 431  
FIG. 432  
FIG. 433  
FIG. 434  
FIG. 435  
FIG. 436  
FIG. 437  
FIG. 438  
FIG. 439  
FIG. 440  
FIG. 441  
FIG. 442  
FIG. 443  
FIG. 444  
FIG. 445  
FIG. 446  
FIG. 447  
FIG. 448  
FIG. 449  
FIG. 450  
FIG. 451  
FIG. 452  
FIG. 453  
FIG. 454  
FIG. 455  
FIG. 456  
FIG. 457  
FIG. 458  
FIG. 459  
FIG. 460  
FIG. 461  
FIG. 462  
FIG. 463  
FIG. 464  
FIG. 465  
FIG. 466  
FIG. 467  
FIG. 468  
FIG. 469  
FIG. 470  
FIG. 471  
FIG. 472  
FIG. 473  
FIG. 474  
FIG. 475  
FIG. 476  
FIG. 477  
FIG. 478  
FIG. 479  
FIG. 480  
FIG. 481  
FIG. 482  
FIG. 483  
FIG. 484  
FIG. 485  
FIG. 486  
FIG. 487  
FIG. 488  
FIG. 489  
FIG. 490  
FIG. 491  
FIG. 492  
FIG. 493  
FIG. 494  
FIG. 495  
FIG. 496  
FIG. 497  
FIG. 498  
FIG. 499  
FIG. 500  
FIG. 501  
FIG. 502  
FIG. 503  
FIG. 504  
FIG. 505  
FIG. 506  
FIG. 507  
FIG. 508  
FIG. 509  
FIG. 510  
FIG. 511  
FIG. 512  
FIG. 513  
FIG. 514  
FIG. 515  
FIG. 516  
FIG. 517  
FIG. 518  
FIG. 519  
FIG. 520  
FIG. 521  
FIG. 522  
FIG. 523  
FIG. 524  
FIG. 525  
FIG. 526  
FIG. 527  
FIG. 528  
FIG. 529  
FIG. 530  
FIG. 531  
FIG. 532  
FIG. 533  
FIG. 534  
FIG. 535  
FIG. 536  
FIG. 537  
FIG. 538  
FIG. 539  
FIG. 540  
FIG. 541  
FIG. 542  
FIG. 543  
FIG. 544  
FIG. 545  
FIG. 546  
FIG. 547  
FIG. 548  
FIG. 549  
FIG. 550  
FIG. 551  
FIG. 552  
FIG. 553  
FIG. 554  
FIG. 555  
FIG. 556  
FIG. 557  
FIG. 558  
FIG. 559  
FIG. 560  
FIG. 561  
FIG. 562  
FIG. 563  
FIG. 564  
FIG. 565  
FIG. 566  
FIG. 567  
FIG. 568  
FIG. 569  
FIG. 570  
FIG. 571  
FIG. 572  
FIG. 573  
FIG. 574  
FIG. 575  
FIG. 576  
FIG. 577  
FIG. 578  
FIG. 579  
FIG. 580  
FIG. 581  
FIG. 582  
FIG. 583  
FIG. 584  
FIG. 585  
FIG. 586  
FIG. 587  
FIG. 588  
FIG. 589  
FIG. 590  
FIG. 591  
FIG. 592  
FIG. 593  
FIG. 594  
FIG. 595  
FIG. 596  
FIG. 597  
FIG. 598  
FIG. 599  
FIG. 600  
FIG. 601  
FIG. 602  
FIG. 603  
FIG. 604  
FIG. 605  
FIG. 606  
FIG. 607  
FIG. 608  
FIG. 609  
FIG. 610  
FIG. 611  
FIG. 612  
FIG. 613  
FIG. 614  
FIG. 615  
FIG. 616  
FIG. 617  
FIG. 618  
FIG. 619  
FIG. 620  
FIG. 621  
FIG. 622  
FIG. 623  
FIG. 624  
FIG. 625  
FIG. 626  
FIG. 627  
FIG. 628  
FIG. 629  
FIG. 630  
FIG. 631  
FIG. 632  
FIG. 633  
FIG. 634  
FIG. 635  
FIG. 636  
FIG. 637  
FIG. 638  
FIG. 639  
FIG. 640  
FIG. 641  
FIG. 642  
FIG. 643  
FIG. 644  
FIG. 645  
FIG. 646  
FIG. 647  
FIG. 648  
FIG. 649  
FIG. 650  
FIG. 651  
FIG. 652  
FIG. 653  
FIG. 654  
FIG. 655  
FIG. 656  
FIG. 657  
FIG. 658  
FIG. 659  
FIG. 660  
FIG. 661  
FIG. 662  
FIG. 663  
FIG. 664  
FIG. 665  
FIG. 666  
FIG. 667  
FIG. 668  
FIG. 669  
FIG. 670  
FIG. 671  
FIG. 672  
FIG. 673  
FIG. 674  
FIG. 675  
FIG. 676  
FIG. 677  
FIG. 678  
FIG. 679  
FIG. 680  
FIG. 681  
FIG. 682  
FIG. 683  
FIG. 684  
FIG. 685  
FIG. 686  
FIG. 687  
FIG. 688  
FIG. 689  
FIG. 690  
FIG. 691  
FIG. 692  
FIG. 693  
FIG. 694  
FIG. 695  
FIG. 696  
FIG. 697  
FIG. 698  
FIG. 699  
FIG. 700  
FIG. 701  
FIG. 702  
FIG. 703  
FIG. 704  
FIG. 705  
FIG. 706  
FIG. 707  
FIG. 708  
FIG. 709  
FIG. 710  
FIG. 711  
FIG. 712  
FIG. 713  
FIG. 714  
FIG. 715  
FIG. 716  
FIG. 717  
FIG. 718  
FIG. 719  
FIG. 720  
FIG. 721  
FIG. 722  
FIG. 723  
FIG. 724  
FIG. 725  
FIG. 726  
FIG. 727  
FIG. 728  
FIG. 729  
FIG. 730  
FIG. 731  
FIG. 732  
FIG. 733  
FIG. 734  
FIG. 735  
FIG. 736  
FIG. 737  
FIG. 738  
FIG. 739  
FIG. 740  
FIG. 741  
FIG. 742  
FIG. 743  
FIG. 744  
FIG. 745  
FIG. 746  
FIG. 747  
FIG. 748  
FIG. 749  
FIG. 750  
FIG. 751  
FIG. 752  
FIG. 753  
FIG. 754  
FIG. 755  
FIG. 756  
FIG. 757  
FIG. 758  
FIG. 759  
FIG. 760  
FIG. 761  
FIG. 762  
FIG. 763  
FIG. 764  
FIG. 765  
FIG. 766  
FIG. 767  
FIG. 768  
FIG. 769  
FIG. 770  
FIG. 771  
FIG. 772  
FIG. 773  
FIG. 774  
FIG. 775  
FIG. 776  
FIG. 777  
FIG. 778  
FIG. 779  
FIG. 780  
FIG. 781  
FIG. 782  
FIG. 783  
FIG. 784  
FIG. 785  
FIG. 786  
FIG. 787  
FIG. 788  
FIG. 789  
FIG. 790  
FIG. 791  
FIG. 792  
FIG. 793  
FIG. 794  
FIG. 795  
FIG. 796  
FIG. 797  
FIG. 798  
FIG. 799  
FIG. 800  
FIG. 801  
FIG. 802  
FIG. 803  
FIG. 804  
FIG. 805  
FIG. 806  
FIG. 807  
FIG. 808  
FIG. 809  
FIG. 810  
FIG. 811  
FIG. 812  
FIG. 813  
FIG. 814  
FIG. 815  
FIG. 816  
FIG. 817  
FIG. 818  
FIG. 819  
FIG. 820  
FIG. 821  
FIG. 822  
FIG. 823  
FIG. 824  
FIG. 825  
FIG. 826  
FIG. 827  
FIG. 828  
FIG. 829  
FIG. 830  
FIG. 831  
FIG. 832  
FIG. 833  
FIG. 834  
FIG. 835  
FIG. 836  
FIG. 837  
FIG. 838  
FIG. 839  
FIG. 840  
FIG. 841  
FIG. 842  
FIG. 843  
FIG. 844  
FIG. 845  
FIG. 846  
FIG. 847  
FIG. 848  
FIG. 849  
FIG. 850  
FIG. 851  
FIG. 852  
FIG. 853  
FIG. 854  
FIG. 855  
FIG. 856  
FIG. 857  
FIG. 858  
FIG. 859  
FIG. 860  
FIG. 861  
FIG. 862  
FIG. 863  
FIG. 864  
FIG. 865  
FIG. 866  
FIG. 867  
FIG. 868  
FIG. 869  
FIG. 870  
FIG. 871  
FIG. 872  
FIG. 873  
FIG. 874  
FIG. 875  
FIG. 876  
FIG. 877  
FIG. 878  
FIG. 879  
FIG. 880  
FIG. 881  
FIG. 882  
FIG. 883  
FIG. 884  
FIG. 885  
FIG. 886  
FIG. 887  
FIG. 888  
FIG. 889  
FIG. 890  
FIG. 891  
FIG. 892  
FIG. 893  
FIG. 894  
FIG. 895  
FIG. 896  
FIG. 897  
FIG. 898  
FIG. 899  
FIG. 900  
FIG. 901  
FIG. 902  
FIG. 903  
FIG. 904  
FIG. 905  
FIG. 906  
FIG. 907  
FIG. 908  
FIG. 909  
FIG. 910  
FIG. 911  
FIG. 912  
FIG. 913  
FIG. 914  
FIG. 915  
FIG. 916  
FIG. 917  
FIG. 918  
FIG. 919  
FIG. 920  
FIG. 921  
FIG. 922  
FIG. 923  
FIG. 924  
FIG. 925  
FIG. 926  
FIG. 927  
FIG. 928  
FIG. 929  
FIG. 930  
FIG. 931  
FIG. 932  
FIG. 933  
FIG. 934  
FIG. 935  
FIG. 936  
FIG. 937  
FIG. 938  
FIG. 939  
FIG. 940  
FIG. 941  
FIG. 942  
FIG. 943  
FIG. 944  
FIG. 945  
FIG. 946  
FIG. 947  
FIG. 948  
FIG. 949  
FIG. 950  
FIG. 951  
FIG. 952  
FIG. 953  
FIG. 954  
FIG. 955  
FIG. 956  
FIG. 957  
FIG. 958  
FIG. 959  
FIG. 960  
FIG. 961  
FIG. 962  
FIG. 963  
FIG. 964  
FIG. 965  
FIG. 966  
FIG. 967  
FIG. 968  
FIG. 969  
FIG. 970  
FIG. 971  
FIG. 972  
FIG. 973  
FIG. 974  
FIG. 975  
FIG. 976  
FIG. 977  
FIG. 978  
FIG. 979  
FIG. 980  
FIG. 981  
FIG. 982  
FIG. 983  
FIG. 984  
FIG. 985  
FIG. 986  
FIG. 987  
FIG. 988  
FIG. 989  
FIG. 990  
FIG. 991  
FIG. 992  
FIG. 993  
FIG. 994  
FIG. 995  
FIG. 996  
FIG. 997  
FIG. 998  
FIG. 999  
FIG. 1000

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like reference numerals represent like parts, and in which:

FIGURE 1 is a block diagram of a graphics system;

FIGURE 2A graphically illustrates a scene;

FIGURE 2B graphically illustrates the relationship between the scene in FIGURE 2A and one example of a higher-level appearance description;

FIGURE 2C graphically illustrates the relationship between the scene and one example of a higher-level appearance description;

FIGURE 3A graphically illustrates a higher-level appearance description that includes several levels of detail;

FIGURE 3B graphically illustrates at least two level of detail traversals of the higher-level appearance description in FIGURE 3A;

FIGURE 3C graphically illustrates another traversal of the higher-level appearance description in FIGURE 3A; and

FIGURE 4 illustrates an example of a method for representing scenes.

DETAILED DESCRIPTION OF THE INVENTION

FIGURE 1 is a block diagram of a graphics system 10. Graphics system 10 includes a host 20 coupled to a graphics system interface 15 which couples to a graphics pipeline 17. Host 20 may be a general or a specific purpose computer and includes a processor 12 and a memory 18, which may include random access memory (RAM) and read only memory (ROM). Specifically, host 20 may be used to execute one or more applications 11 having image graphics and visualization software that may be stored in memory 13 and/or an input/output device 14. Results may be displayed using display 90 and/or stored in input/output device 14, which may be any suitable storage medium. Data processing may be performed using special purpose digital circuitry contained either in host 20 or in a separate device. Such dedicated digital circuitry may include, for example, application-specific integrated circuitry (ASIC), state machines, fuzzy logic, as well as other conventional circuitry. Host 20 may also include a portion of a computer adapted to execute any of the well known MS-DOS, PC-DOS, OS2, UNIX, MAC-OS, and Windows operating systems or other operating systems including nonconventional operating systems. Host 20 may also be coupled to a communication link 16 that may be connected to a computer network, a telephone line, an antenna, a gateway, or any other type of communication link.

Interface 15 may be any software graphics or firmware interface such as OpenGL® or DIRECT3D® that includes procedures and functions and that may be used to control low-level operations in graphics pipeline 17. In operation, interface 15 is operable to control the processing of image data in graphics pipeline 17 in response to selected commands that are passed from an application 11 such as a programmable shader. Data is passed through some or all of the elements in graphics pipeline 17 and may then be transferred from frame buffer 70 to display 90 for viewing. For example, pixels may be written to and read from frame buffer 70 using OpenGL® function calls such as the DrawPixels and ReadPixels command, and the function CopyPixels can be used to copy a block of pixels from one region of frame buffer 70 to another.

More specifically, graphics pipeline 17 includes a vertex operations module 30 and a pixel operations module 40. Vertex operations module 30 and pixel operations module 40 are each coupled to a rasterization hardware 50. Rasterization hardware 50

is coupled to a frame buffer operations module 60, which in turn is coupled to a frame buffer 70. Frame buffer 70 may couple to pixel operations module 40. Pixel operations module 40 is also coupled to a texture memory 80, which is also coupled to rasterization hardware 50. Graphics pipeline 17 may include software, firmware, hardware, or a combination thereof. Interface 15 may be a standalone module, reside on host 20, or a combination thereof.

Because interfaces 15 such as OpenGL® are procedurally based, graphics pipeline 17 performs those low-level operations on all of the pixels passed in response to the OpenGL® procedure or function call. On the other hand, in most cases it is desirable to perform various operations on a variety of pixels to achieve the appearance the artist desires. One method for performing these various operations on a variety of pixels is to employ several passes through graphics pipeline 17 ( a multi-pass scenario), each using a unique operation and/or variety of pixels.

Thus, it may be helpful to illustrate a single pass through graphics pipeline 17. Host 20 sends image data to pixel operations module 40, which may utilize a lookup table to apply a scale or bias such as a color contrast or brightness to pixels passed thereto. Host 20 also sends geometry data to vertex operations module 30. The geometry data usually includes texture coordinates or vertices (s,t,r,q) that are projected points that correspond to a location (x,y,z,w) in an image plane. The geometry data may also include normals at each of these vertices for each of the three channels (usually red, green, and blue). Vertex operations module 30 transforms image data from pixel operations module 40 into a raster coordinate system. Usually, this includes tessellation, or breaking down a continuously smooth surface into triangular surfaces. Rasterization hardware 50 usually interpolates the tessellated vertices to populate the pixels within each of these surfaces. In some applications, rasterization hardware 50 may also request a texture map from texture memory 80 which is then applied to all of the pixels in rasterization hardware 50. These pixels are then passed to frame buffer 70.

Frame buffer operations module 60 then may perform a variety of functions on the data passed from rasterization hardware 50 and then pass this data to frame buffer 70. Some of these functions include, but are not limited to, a depth test, stencil test, and blending, and are performed on all of the pixels passed to frame buffer operations

module 60. A depth test typically discards portions of an image region that fail a depth comparison. For example, the depth test may be used to clip surfaces that are further from, or are obstructed by, an object that is nearer in a field of view. A stencil test may be used as an arbitrary comparison that allows selected pixels to be rejected based on the outcome of a comparison between the value in the stencil buffer and the reference value, usually an integer. Blending usually includes operations that may be performed on the pixels in the frame buffer, such as adds, subtracts, multiplies, or clears, and is typically used when assigning color values to pixels. An operation may be performed for each of the three color channels. When frame buffer 70 has performed this operation on all of the pixels, the pixels are usually sent to a display 90.

Where programmable applications 11 such as shading algorithms are used to model the appearance of objects, an artist typically describes the appearance of one or more portions of an image by selecting those pixels that should be altered. For example, a programmable shading algorithm may be used to provide various atmospheric, light, shading, surface details, textures, and/or colors. These functions may parameterize the appearance of selected objects.

These complex appearance effects typically result in different operations being performed on each resultant geometry-based vertex. One example may be a three-D lighting operation that models the diffuse reflection of colored, directional light sources from colored surfaces. Algorithms may use an illumination function that calculates the diffuse reflection of colored directional light for each vertex of a colored surface. For example, the illumination function for a single vertex is a vector dot product of the light source coordinates and the vertex normal, multiplied by the light color, the vertex material color, and the attenuation.

Application 11 may process one or more pixel-based portions of an image for a given geometry-based vertex by passing selected portions of image data through graphics pipeline 17 multiple times with different parameters. This allows interface 15 such as OpenGL® to be used as a single-instruction, multiple-data (SIMD) computing surface by using several basic OpenGL® functions in multi-pass operations that are called by application 11. One such function may include, for example, CopyTexImage which may define a texture array from frame buffer 70.



One such application 11 that may utilize interface 15 as a SIMD computing surface is one that utilizes the RenderMan shading language. Details for translating a shading language such as RenderMan into multiple passes through a graphics pipeline 17 driven by a graphics interface 15 such as OPENGL® may be found in co-pending  
5 U.S. Patent Application serial number 09/056,583, entitled "System and Method for High-Speed Execution of Graphics Application Programs Including Shading Language Instructions", filed April 8, 1998.

Programmable shaders are typically descriptive, rather than procedural like an interface 15 such as OPENGL®. The invention contemplates the use of any descriptive programmable application 11 that uses multiple passes to alter the appearance of an image scene, or one or more objects therein, in order to obtain a final image. These applications improve the efficiency and flexibility of the artistic visualization process. For example, application 11 may be used in reflective mapping (e.g., where a surface appears shiny and reflects surrounding surfaces). Application  
10 11 may also perform bump mapping, which typically define normal perturbations that may be added to a surface normal; shadow depth maps typically seen from a particular viewpoint, in addition to programmable shaders. These descriptive applications are desirably implementation-independent, and provide a desired result.

Each application 11 may reside on host 20 or may be one or more separate modules. Any data upon which application software may operate, including scenes and any objects therein, may be referred to as image data. This image data may originate from memory in host 20, input/output device 14, and/or in a separate storage medium (not explicitly shown). Application 11 and image data residing on host 20 are used to illustrate one aspect of the invention. Interface 15 may couple host 20 to  
15 graphics pipeline 17 in some embodiments or couple a separate application program 11 to host 20 in others.

Higher-level or abstract descriptions 13a – 13n may be used as an infrastructure between interface 15 and an application 11 to provide a framework for representing the appearance of an image scene, or objects therein. Higher-level or  
20 abstract descriptions 13a – 13n may reside on host 20 (for example, as a part of an application 11 or in a graphics library (not explicitly shown)), or may be one or more separate modules. Each higher-level appearance description 13a - 13n may also be an  
25

application 11 such as a programmable shading algorithm, and includes a data structure to maintain and/or access data such as geometry and appearance parameters and may be any hierarchical scene description including, but not limited to, scene graphs or retained mode interfaces. These appearance parameters include a description and an appearance of objects within the described scene. In addition, higher-level appearance descriptions 13a - 13n may be used as a control interface to preside over transformations such as library procedures or functions that are used to build up scene graphs. For example, higher-level appearance descriptions 13a - 13n may be used to cull or manage different levels of details in a traversal, and/or to recharacterize parameters so that bandwidth, memory and/or processor resources may be minimized.

Higher-level appearance descriptions 13a - 13n may be written, for example, as programmable shaders in which a higher-level appearance description may be imbedded. For example, higher-level appearance description 13a may be traversed at least one time to produce a final abstract representation or description 13b. Alternatively, a plurality of higher-level appearance descriptions 13b - 13n-1 may be traversed to create a final abstract representation or description 13n. Traversing may be implemented using a variety of methods, and includes translating or mapping a higher-level appearance description into another representation. For example, the final abstract representation represents a set of multi-pass steps or operations that may be directly executed by using interface 15. As described below, the order of these steps may be optimized to expedite processing or execution time. Thus, interface 15 and/or graphics pipeline 17 may use the final abstract representation to render or draw the described scene.

FIGURES 2A-2C graphically illustrate the relationship between a scene and two examples of higher-level appearance descriptions. A scene S is illustrated in FIGURE 2A, includes a plurality of image pixel values, and may be described by a variety of appearance descriptions. In this embodiment, scene S is a portion of image data. Within scene S, the three spheres  $S_1$ - $S_3$  each have a radius of 1, and are located at respective (x,y,z) coordinates (0,0,0), (0,2,0), and (0,4,0).

Referring to FIGURES 2B and 2C, scene graphs or retained mode interfaces may be used as one example of higher-level appearance descriptions for the

appearance of three spheres in scene S of FIGURE 2A. Scene graphs are used to illustrate several aspects of the invention. For example, scene graphs may be used by visualization applications 11 as a data structure to retain data, as well as an infrastructure to control geometry and other transformations, procedures, functions or libraries to build up the scene graph. Each scene graph typically includes an internal node, where a transformation or function may be applied to the shape, appearance and/or geometry of all of its children. Any kind of transformation or function may be applied, including simple functions such as translations, and complex functions such as blending. Each scene graph also includes at least one leaf node, which has no children. Each node typically includes some shape, form, or geometry description that may be used, such as a sphere, and an appearance, such as radius = 1, and/or color = red. For example, the shape may be a sphere of radius one whose center is at (0,0,0). This sphere may be translated two units in the y direction, so that its center is at a resultant position (0,2,0). Two examples of scene graphs are discussed in conjunction with FIGURES 2B and 2C.

FIGURE 2B illustrates a scene graph 100 that may be used to represent scene S. Scene graph 100 has three shape nodes and two transformation nodes. In this illustration, each shape node has an identical appearance and geometry (e.g., a sphere with radius 1). Scene graph 100 illustrates an internal node with one leaf child, shape leaf node 101, where the sphere with radius 1 is at the origin (0,0,0). Scene graph 100 may then be traversed to reach another internal node 102, a transformation node that has one leaf child, shape leaf node 103. In this illustration, transformation node 102 translates shape leaf node 103 two units in the y direction. Thus, the sphere represented by shape leaf node 103 is at a location (0,2,0). The third sphere may be obtained by traversing scene graph 100 through another internal transformation node 104 to reach its child, shape leaf node 105.

Similarly, FIGURE 2C illustrates a scene graph 200 that also represents scene S. Scene graph 200 has three shape leaf nodes 201, 202, and 203. In this illustration, each of the three shape nodes in scene graph 200 has a different geometry. For example, shape node 201 is a sphere with radius 1 located at (0,0,0). On the other hand, shape nodes 202 and 203 are each spheres with radius 1 located at respective locations (0,2,0) and (0,4,0).

Scene graphs 100 and 200 may be implemented by a variety of techniques and as a variety of data structures. For example, scene graphs may be implemented in an object-oriented language, where each node may be represented as an object. In addition, each traversal may be implemented as a procedure or function. For example, in object-oriented programming, the procedure may begin at any node within a scene graph, and each object may be programmed to know how to traverse, shade, tessellate, and/or draw itself. For example, an interior node may traverse through all of its children until it is completed.

Values within scene graphs 100 and 200 may be used to manage user data and resources, such as removing or reducing data that is not visible or otherwise not required in a scene. For example, an object deep in the background of the scene may be visible in one or more perspective angles of the scene. Similarly, a level of detail traversal may be associated with a level of detail that will appear in the scene. For example, foreground objects often are illustrated with a higher level of detail than those objects in the background of a scene. Such advantages minimize or reduce unnecessary processing time and/or resources that are required to process this data, and are further discussed in conjunction with FIGURES 3A and 3B.

Similarly, values within scene graphs 100 and 200 may also be structured so that parameters may be supplied and evaluated on scene geometry for a variety of pixel-based applications, including programmable shading. These parameters may then be accessed by frame buffer 70 for multi-pass operations in graphics pipeline 17. Referring again to FIGURE 2A, parameters  $P_{v1}$ - $P_{v4}$  define values of a selected parameter of the geometry. For example, a sphere may include parameters  $P_{v1}$ - $P_{v4}$  such as colors or blends thereof that vary across its surface. Any appearance-related parameters for each vertex within the geometry may be derived from parameters  $P_{v1}$ - $P_{v4}$ .

Each scene graph may retain a list of parameters for a parametric surface. For example, traversals of scene graphs 100 and 200 may also be used to generate new scene graphs (not explicitly shown), which parameterize parameters  $P_{v1}$ - $P_{v4}$  for the parametric surface defined by scene S. These new scene graphs may retain the list of parameters for scene S. Any surface may be reparameterized by using a number of points that uniquely define the surface, such as three points for a sphere and four

points for any other surface. For example, a parametric description for sphere of radius  $r=1$  whose center is at  $(0,0,0)$  is given by the equation  $x^2 + y^2 + z^2 = 1$ . In order to provide correspondence between parameters  $P_{v1}$ - $P_{v4}$  to geometry, points on the sphere may be found to satisfy the geometric requirements. This provides the variation of each parameter as a function of surface coordinates, which may be drawn as a vertex color, or set of colors. For example,  $x$  may be written or reparameterized to the value  $x = r \sin \theta \cos \phi$ ;  $y = r \sin \theta \sin \phi$ ; and  $z = r \cos \theta$ . Reparameterization also may provide derivatives of the parameters, such as rates of change in a shader across the surface.

These parametric equations may be evaluated or tessellated by graphics pipeline 17 to approximate the surface. If the parameter is changing, as graphics pipeline 17 tessellates the surface, the reparameterization may provide the value at each vertex of the geometry for parameters  $P_{v1}$ - $P_{v4}$ . On the other hand, if the parameter is constant, only one value need be queried. In some applications, it may be desirable to cache results in a storage medium such as texture memory 80 from tessellations performed within a first pass through graphics pipeline 17. Such a configuration may save processing time and resources. These values may be then used by an interface 15 such as OPENGL® to draw the surface. These new scene graphs may represent a final representation from which the desired scene may be drawn, or further traversals may be performed to reach an appropriate level of detail and to generate the final representation.

FIGURES 3A-3C graphically illustrate another aspect of a higher-level appearance description. Scene graphs are used as one example of a higher-level appearance description to illustrate several aspects of the invention.

For example, FIGURE 3A illustrates a scene graph 300 that includes objects A, B, C, and D. Each object may be a function, an application 11, an image scene, or object therein, and includes corresponding parameters  $V_A - V_D$ . Each of the parameters may represent geometry, texture, color, etc. Scene graph 300 also includes a plurality of appearances each associated with a level of geometry desirable for its representation in the scene. Here, an artist has designated three different levels of representation  $A_1$ - $A_3$  for an object A, e.g., for near, mid-range and distant

perspectives, respectively. These representations may include a number of tessellated surfaces that provide sufficient detail of the appearance as desired.

To illustrate, near representation  $A_1$  may be tessellated into 10,000 surfaces; mid-range representation  $A_2$  may be tessellated into 1,000 surfaces; and distant representation  $A_3$  may be sufficiently described with 100 tessellated surfaces. These representations may be defined by the user or artist, or automatically defined. A rendering of shading for representation  $A_1$  may require, for example, 5,000 passes through graphics pipeline 17, whereas rendering for shading for representation  $A_3$  may require 10 passes. Such a configuration desirably minimizes data sent to graphics pipeline 17, because only those tessellated surfaces that are necessary to achieve a desired appearance are used. Such an advantage also may reduce resources such as memory, processing, and/or bandwidth requirements.

FIGURE 3B graphically illustrates at least two level of detail traversals of the higher-level appearance description in FIGURE 3A. Scene graph 310 may represent a final representation from which the desired scene may be drawn, or further traversals may be performed to reach an appropriate level of detail and to generate the final representation. This traversal reduces processing time and/or resources that would otherwise be needed to process and/or clip objects not in the scene.

To illustrate, an appropriate representational level for each parameter or object A may be selected by a user request. Many methods may be used to implement such a selection process. For example, an artist may manually, or a flight simulator application may automatically, select near representation  $A_1$  and traverse scene graph 300 to generate a second scene graph 310. For example, the artist may select  $A_1$  to be in a foreground of a scene. On the other hand, the flight simulator may algorithmically calculate the relative position and/or size of an object as a function of movement during execution of the program.

As another illustration, the traversal may represent an algorithmic (e.g., shading) level of detail traversal. For example, an artist may desire that object A is diffuse and specular for different representations  $A_1$  and  $A_2$ . Distances to object A within the scene may be similarly calculated as discussed above, and based on the expected level of detail desired, the scene graph may select the shading algorithm to apply to object A.

FIGURE 3C graphically illustrates another traversal of the higher-level appearance description in FIGURE 3A. The total number of temporary requirements for a storage medium such as texture memory 80 in graphics pipeline 17 includes an amount of space allotted for parameters  $V_A - V_D$ . Scene graph 300 may be traversed to generated scene graph 320 to manage and thus minimize these storage requirements and/or to avoid or eliminate data replication. For example, scene graph 300 may be translated into scene graph 320, a form that includes new variable name  $V_{new}$ . Variable  $V_{new}$  may be shared between objects and/or algorithms A-D. These variable names will not collide during processing within graphics pipeline 17, because they are each typically used in a multi-pass scenario at separate times. This provides the advantage of managing resources that may be statically allocated.

To illustrate, parameters  $V_A - V_D$  may each represent a color of corresponding object A - D. Thus for scene graph 300, object A may be processed through graphics pipeline 17 and assigned its color  $V_A = \text{red}$ . Similarly and at different processing times, object B may be processed through graphics pipeline 17 and assigned its color  $V_B = \text{blue}$ , and so on. The color may be blended with the pixels that represent each object by retrieving the associated object's parameter from texture memory 80. In contrast, scene graph 320 may be processed by using only a single variable  $V_{new}$  that is reassigned for each object as it is processed through graphics pipeline 17. Scene graph 320 may represent a final representation from which the desired scene may be drawn, or further traversals may be performed to reach an appropriate level of detail and to generate the final representation.

The higher-level appearance descriptions described and discussed in conjunction with FIGURES 2A-3C may be used in any combination, for any number of traversals. For example, culling and/or level of detail traversals may be performed in conjunction with parameterization.

FIGURE 4 illustrates an example of a method for representing scenes. Although steps 400-406 are illustrated as separate steps, various steps may be ordered in other logical or functional configurations, or may comprise single steps. The method may be used with any hierarchical way to describe a scene, including scene graphs or retained mode interfaces. Again, a scene graph is used to illustrate one aspect of the invention.

The method begins in step 400, where a first scene graph is created with one or more applications 11. As discussed in conjunction with FIGURES 2A-3C, the scene graph may be implemented by a variety of different methods or algorithms, and includes a description of the appearance and geometry of objects within a scene.

5 In step 402, a request for action is generated, either by a user such as an artist, or automatically, by an application 11 such as a flight simulator. For example, the user may requests that the scene be drawn or rendered which may be represented by a request to traverse the first scene graph. In response to the user request, at least one traversal of the scene graph is performed to create a new scene graph in step 404. In some applications, where a plurality of applications 11 are in use, the request may be to perform a traversal over all applications 11.

10 This traversal may facilitate defining temporary storage requirements in step 406, which may manage and thus minimize resources that may be statically allocated. For example, temporary requirements for a storage medium such as texture memory 80 may be minimized by translating a scene graph into a form that utilizes shared variables, as was discussed in conjunction with FIGURE 3C. These shared variables will not collide during the rendering process, because each of these variable names is typically used in a multipass scenario at separate times.

15 In step 408, if no additional data management is to be performed, the new scene graph is the final scene graph, which is ready to be drawn by graphics pipeline 17. The method proceeds to step 410, where the final scene graph may be rendered in response to the user request in step 402. On the other hand, the action requested in step 402 may include a number of traversals to be performed and new scene graphs generated, in order to reach a final scene graph. Thus, the method proceeds to step 20 412 and then returns to step 404 to traverse the prior scene graph and to generate a new scene graph in response to the traversal.

25 For example, the method may determine in step 408 that surface parameterizations and/or level of detail traversals are to be performed, as discussed in conjunction with FIGURES 2C-3C. In addition, additional scene graphs to be used in multiple applications 11 may be traversed to optimize the use of variables therefor.

30 Thus, it is apparent that there has been provided in accordance with the present invention, a system and method for representing scenes that satisfies the advantages



set forth above. For example, the present invention provides an infrastructure for managing data to control scene appearances that may minimize necessary computing time and resources. The invention allows for surface parameters to be retained and evaluated in a multi-pass scenario using a graphics system interface such as  
5 OPENGL®. While the invention has been particularly shown by the foregoing detailed description, various changes, substitutions, and alterations may be readily ascertainable by those skilled in the art and may be made herein without departing from the spirit and scope of the present invention as defined by the following claims.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25